HAR Machine Learning

Patrick Siu

January 25, 2016

Background

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it.

Project Goal

We will use public data sourced from http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) to evaluate accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The project will attempt to predict the manner in which the exercise was performed.

Data Loading

```
rawDataFile_training <- "pml-training.csv"
downloadURL_training <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"

rawDataFile_test <- "pml-testing.csv"
downloadURL_test <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

##Ensure raw data file exists
if(!file.exists(rawDataFile_training)){
    download.file(downloadURL_training, rawDataFile_training)
}
if(!file.exists(rawDataFile_test)){
    download.file(downloadURL_test, rawDataFile_test)
}

##Load raw data
training <- read.csv("pml-training.csv", na.strings = c("NA", "", "#DIV/0!")) #Converting invalid a
nswers to NA
test <- read.csv("pml-testing.csv", na.strings = c("NA", "", "#DIV/0!"))
```

Exploratory Analysis

```
dim(training)
```

```
## [1] 19622   160
```

```
str(training$classe)
```

```
##  Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
dim(test)
```

```
## [1]  20 160
```

Notes: Within the training set, the “classe” column is the outcome that we want to predict. The test set has the same number of columns, but the classe column is replaced by problem_id.

Tidy Data Set

```
#Remove unnecessary columns
remove_fields <- c("X", "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2", "cvtd_timestam
p", "new_window", "num_window", "problem_id")
training <- training[, !(names(training) %in% remove_fields)]
test <- test[, !(names(test) %in% remove_fields)]  # Apply same treatment to test set

sum(complete.cases(training))  # 0 complete cases!
```

```
## [1] 0
```

```
sum(complete.cases(test))      # 0 complete cases!
```

```
## [1] 0
```

```
#Using summary(training), we can see that there are many columns that have 19216 NAs out of 19622 o
bservations. All other columns are complete. We will exclude the columns with NA.
training <- training[,colSums(is.na(training)) == 0]
test <- test[,colSums(is.na(test)) == 0]

dim(training); dim(test)
```

```
## [1] 19622    53
```

```
## [1] 20 52
```

```
nsv <- nearZeroVar(training, saveMetrics=TRUE) #Near Zero Var check shows false for all columns
```

Tidy data set ready for analysis. All NAs purged and relevant columns selected. Training set has one additional column “classe” as the outcome.

Partitioning the data

Given that we are assigned with only 20 rows of test data (0.1%) compared to 19,622 rows of data in the training set, we have an uneven distribution for testing. Realistically speaking, this test set is actually an evaluation set. Therefore we will split the training set for 60% training and 40% validation. We use the term validation instead of test to avoid confusion with the ‘evaluation’ test set that is given.

This 40% hold out “validation” set is required in order to obtain OOSE (Out of Sample Error)

5 k-fold cross-validation will be used instead of bootstrapping for performance tradeoff.

```
set.seed(628)  #Reproducibility
inTrain <- createDataPartition(y=training$classe, p=0.6, list=FALSE)

validation <- training[-inTrain,]
training <- training[inTrain,]  ## Note:  Destructive edit
dim(training)
```

```
## [1] 11776    53
```

```
dim(validation)
```

```
## [1] 7846   53
```

Training the Model

We will run three different models and select the one with the highest accuracy.

```
fitControl <- trainControl(method = "cv",
                           number = 5,
                           allowParallel = TRUE)

#Recursive Partitioning for Classification, Rgression, and Survival Trees (RPART)
modFit_rpart <- train(classe ~ .,method="rpart",data=training)
```

```
## Loading required package: rpart
```

```
print(modFit_rpart$finalModel)
```

```
## n= 11776
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 11776 8428 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130.5 10796 7452 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -34.35 944    2 A (1 0.0021 0 0 0) *
##      5) pitch_forearm>=-34.35 9852 7450 A (0.24 0.23 0.21 0.2 0.12)
##       10) magnet_dumbbell_y< 439.5 8325 5967 A (0.28 0.18 0.24 0.19 0.11)
##         20) roll_forearm< 126.5 5264 3130 A (0.41 0.18 0.19 0.16 0.064) *
##         21) roll_forearm>=126.5 3061 2055 C (0.073 0.18 0.33 0.24 0.19) *
##       11) magnet_dumbbell_y>=439.5 1527  739 B (0.029 0.52 0.046 0.22 0.18) *
##    3) roll_belt>=130.5 980    4 E (0.0041 0 0 0 1) *
```

```
pred_rpart <- predict(modFit_rpart, validation)
validation$predRight <- pred_rpart==validation$classe

accuracy_rpart <- as.numeric(confusionMatrix(validation$classe, pred_rpart)$overall[1])
round(accuracy_rpart * 100, 2)
```

```
## [1] 49.22
```

```
#Random Forest (RF)
#modFit_rf <- train(classe~ .,data=training,method="rf",prox=TRUE)
modFit_rf <- randomForest(classe ~ ., data = training, trControl = fitControl)  #randomForest packa
ge significantly faster than caret package
print(modFit_rf)
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training, trControl = fitControl)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.59%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3347    1    0    0    0 0.0002986858
## B   12 2262    5    0    0 0.0074594120
## C    0   13 2039    2    0 0.0073028238
## D    0    0   28 1901    1 0.0150259067
## E    0    0    4    4 2157 0.0036951501
```

```
pred_rf <- predict(modFit_rf, validation)
validation$predRight <- pred_rf==validation$classe

accuracy_rf <- as.numeric(confusionMatrix(validation$classe, pred_rf)$overall[1])
round(accuracy_rf * 100, 2)
```

```
## [1] 99.18
```

```
#Generalized Boosted Regression Models (GBM)
modFit_gbm <- train(classe ~ ., method="gbm",data=training,verbose=FALSE, trControl = fitControl)
print(modFit_gbm)
```

```
## Stochastic Gradient Boosting
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 9422, 9420, 9421, 9421, 9420
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa      Accuracy SD
##   1                   50      0.7556908  0.6902754  0.012374985
##   1                  100      0.8192941  0.7713162  0.004834249
##   1                  150      0.8519039  0.8125801  0.008189545
##   2                   50      0.8529226  0.8136862  0.005950939
##   2                  100      0.9075250  0.8829603  0.005682413
##   2                  150      0.9322357  0.9142444  0.004129917
##   3                   50      0.8977604  0.8705292  0.008586705
##   3                  100      0.9414067  0.9258643  0.002944014
##   3                  150      0.9590699  0.9482207  0.002830837
##   Kappa SD
##   0.015517551
##   0.006187277
##   0.010373156
##   0.007528490
##   0.007174858
##   0.005220093
##   0.010891538
##   0.003720145
##   0.003582304
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
pred_gbm <- predict(modFit_gbm, validation)
validation$predRight <- pred_gbm==validation$classe

accuracy_gbm <- as.numeric(confusionMatrix(validation$classe, pred_gbm)$overall[1])
round(accuracy_gbm * 100, 2)
```

```
## [1] 95.59
```

```
stopCluster(cluster)
```

Based on validation data set:

- RPART accuracy 49.22%
- RF accuracy 99.18%
- GBM accuracy 95.59%

Select Final Model and Predict on Test Set

Random Forest model has the highest accuracy at 99.18%. We will use this as our final model.

```
pred <- predict(modFit_rf, test)
print(pred) #Final results
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  B  A  A  E  D  B  A  A  B  C  B  A  E  E  A  B  B  B
## Levels: A B C D E
```

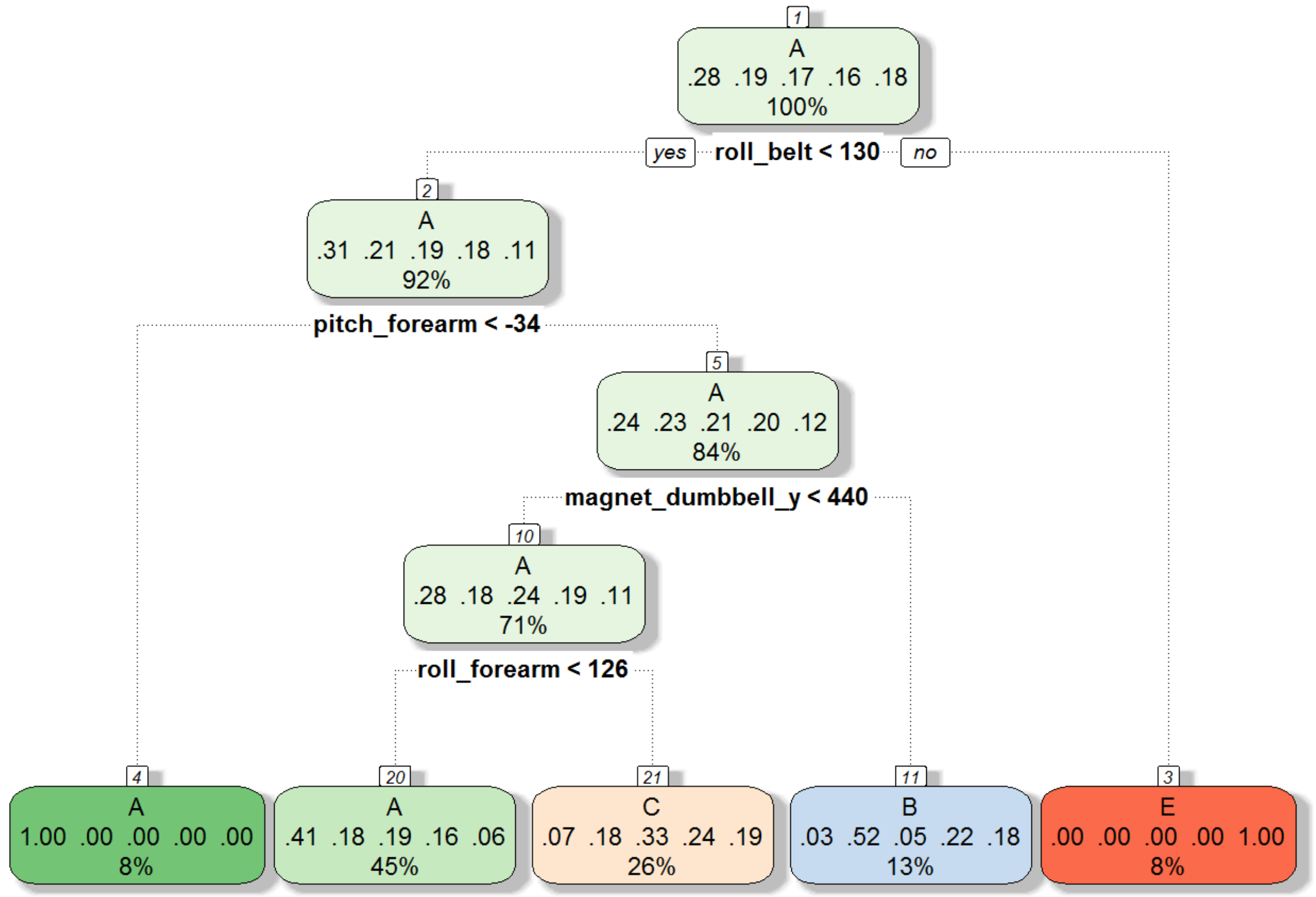
Final Model Accuracy: 99.18%

Final Model Out of Sample Error: 0.82%

Appendix

A quick visualization using rpart. Relatively low accuracy but easy interpretability

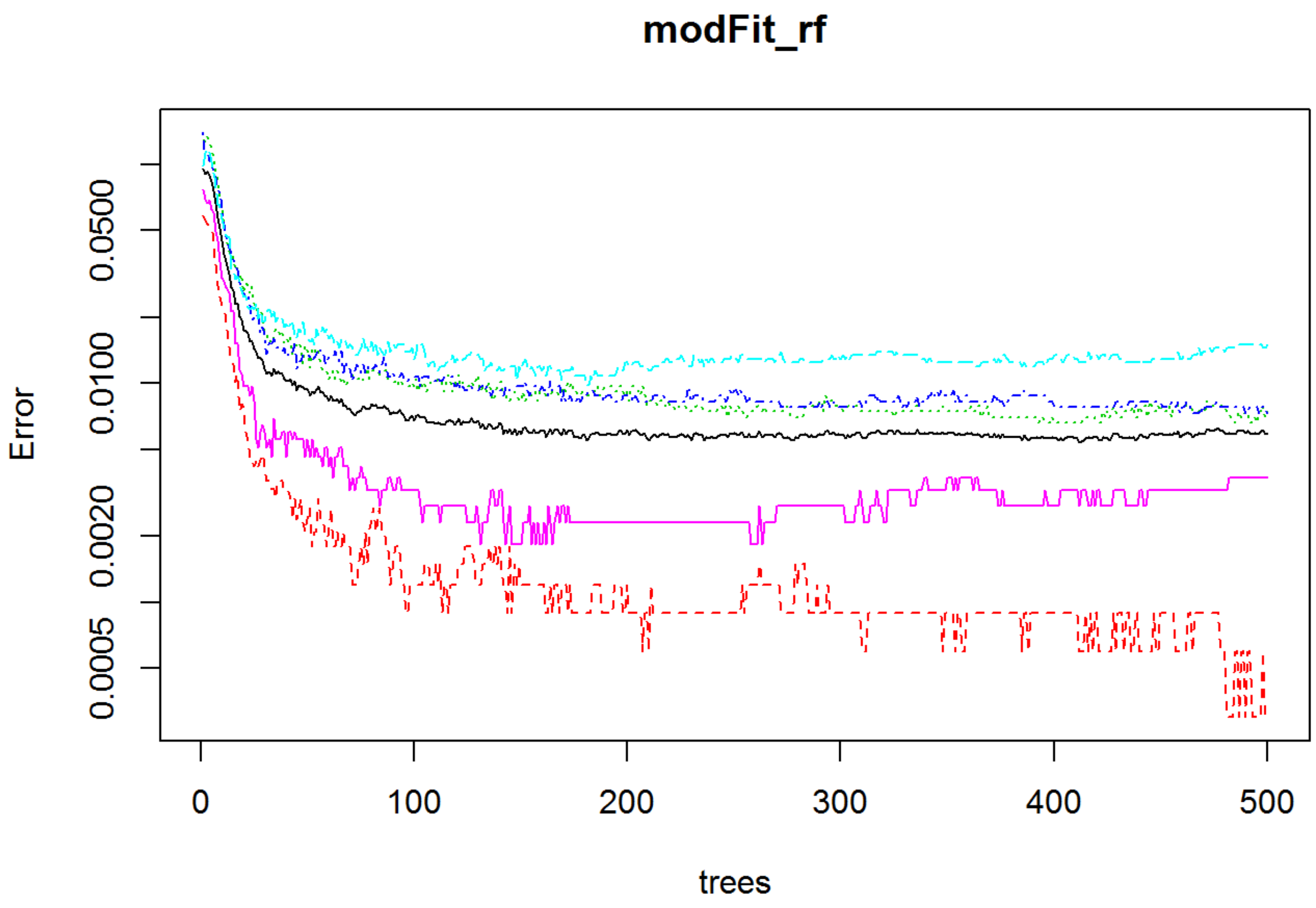
```
fancyRpartPlot(modFit_rpart$finalModel)
```



Rattle 2016-Jan-31 08:58:56 siup

[Random Forest] error rates over number of trees

```
plot(modFit_rf, log="y")
```


modFit_rf
Error
0.0500
0.0100
0.0020
0.0005
0
100
200
300
400
500
trees